Javascript avancé
Module 4 – TypeScript

OBJECTIFS

Appréhender les bases du TypeScript

Utiliser la POO avec Typescript

Être capable d'utiliser les modules et génériques

SOMMAIRE

Introduction au Typescript

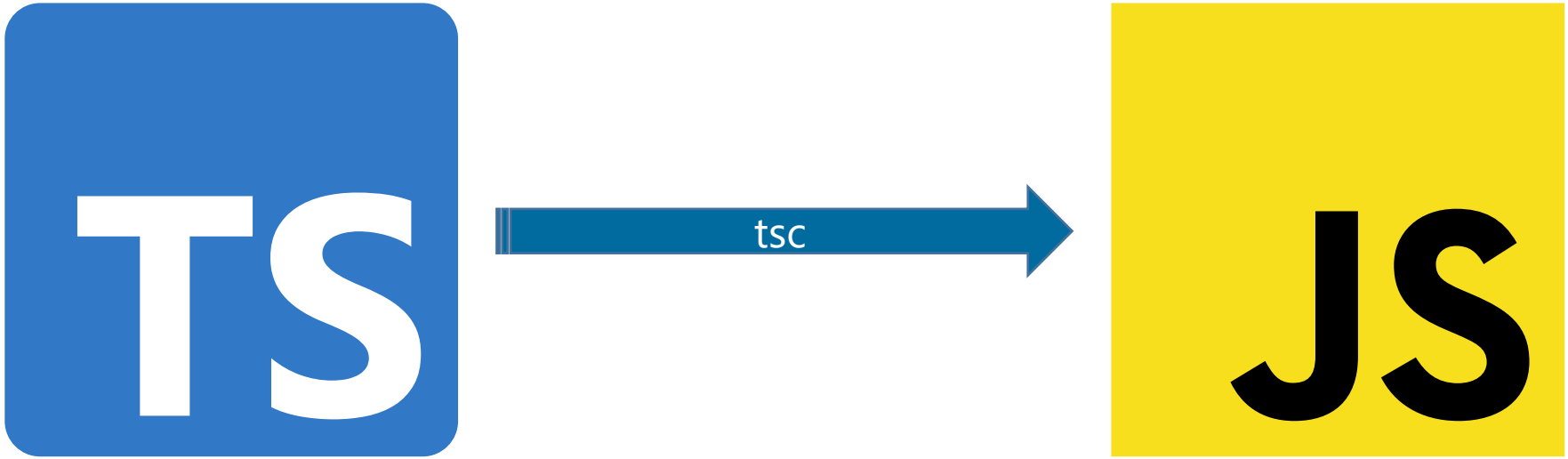
Curriculum vitae

- 2012
- https://www.typescriptlang.org
- TypeScript is a typed superset of JavaScript that compiles to plain Javascript
- Any browser, any host, any OS, open source

Pourquoi Typescript

- Typage statique
- Utilisation des classes
- Utilisation des interfaces
- Utilisation des génériques
- Utilisation des énumérés
- Utilisation des modules

Transcompilation




```
npm install -g typescript
tsc index.ts
```

DÉMONSTRATION

Installation de TypeScript

Les variables

Variables

```
let estAdministrateur: boolean = true;
let age: number = 39;
let profession: String = "Développeuse full stack";

let aEviter = 42;
let meilleur: any = 42;
meilleur = "Quarante deux";
```

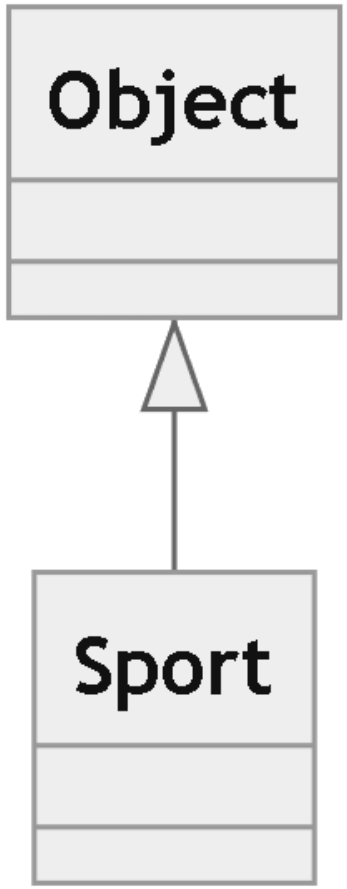
```
enum Etablissement {Maternelle, Primaire, College, Lycée, Fac};
let monEtablissement: Etablissement = Etablissement.Maternelle;
```

DÉMONSTRATION

Variables et énumérés

La POO

Les classes

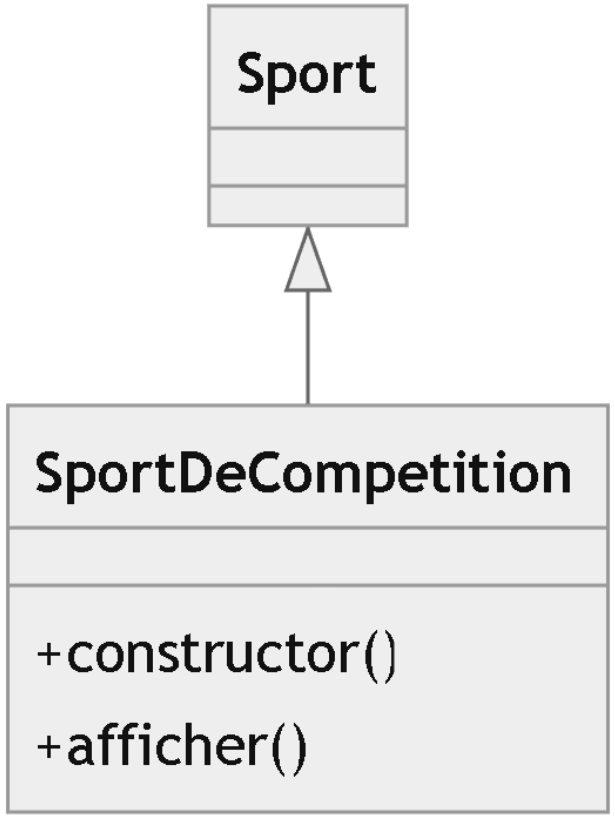



```typescript
class Sport {
    public nom: string;
    private prive: string;

    constructor(
        nom: string,
        public description: string
    ) {
        this.nom = nom;
        this.prive = "valeur privée";
    }

    public afficher(): void
        console.log(this.nom + ' ' + this.description);
    }
}
```

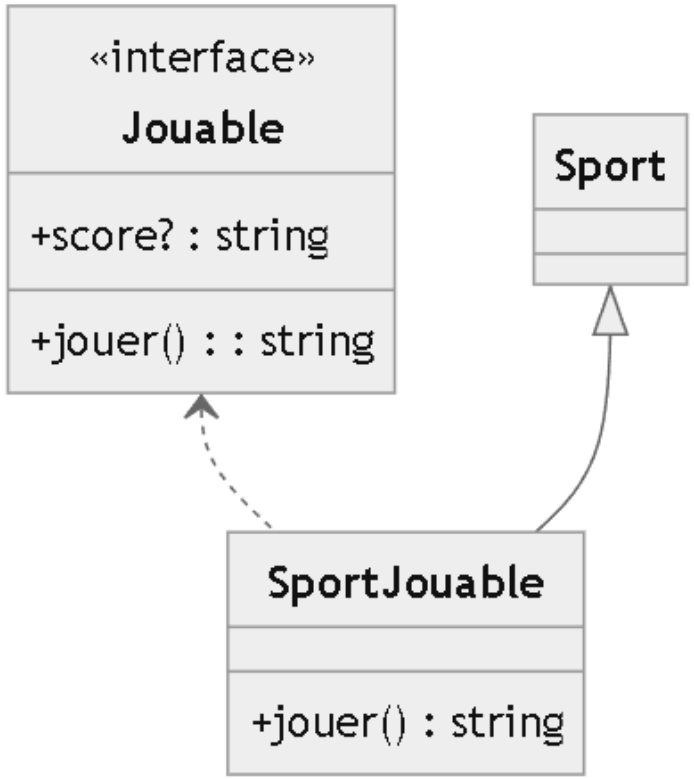
Héritage




```
class SportDeCompetition extends Sport {
    constructor(
        nom: string,
        description: string,
        public niveau: string
    ) {
        super(nom, description);
    }

    public afficher(): void {
        super.afficher();
        console.log(("Niveau : " + this.niveau));
    }
}
```

Interface




```typescript
interface Jouable {
    score?: string;

    jouer(joueur1: string, joueur2: string): string;
}

class SportJouable extends Sport implements Jouable {
    jouer(joueur1: string, joueur2: string): string {
        return '';
    }
}
```

Interface de méthode

```
interface jouer {
    (joueur1: string, joueur2: string): string;
}

let jouerAuBadminton: jouer = (joueur1, joueur2) => {
    return 'On joue au badminton';
}
```

DÉMONSTRATION

Classe, héritage et interface

Les génériques

Les génériques

Les tableaux :

```
let tableauType: number[] = [1, 2, 3];
let tableauGenerique: Array<number> = [9, 8, 7];
```

Les classes :

```
class Viande {
}

class Legume {
}

class Sandwich<V extends Viande, L extends Legume> {
    constructor(
        public viande: V,
        public legume: L
    ) {
    }
}
```

DÉMONSTRATION

Les génériques

Les modules

Modules

```
module BusinessObject {
    export class Sport {
    }
}

let sportModule: BusinessObject.Sport;

import BO = BusinessObject;
let sportAvecModule: BO.Sport;
```

TRAVAUX PRATIQUES

Vente aux enchères

Conclusion

- Vous connaissez les bases du langage Typescript
- Vous savez créer des classes, interfaces en Typescript
- Vous avez connaissance de l'utilisation des génériques